EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	21882040	@ad<"20020118"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:50
L2	30306	"711"/$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:50
L3	1108	"711"/147.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:50
L4	47	allocat$4 near (first adj process)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:56
L5	38	allocat$4 near (second adj process)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L6	47	allocat$4 near (first adj process)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L7	38	allocat$4 near (second adj process)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L8	15	L6 and L7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L9	2	"20020059503".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57

EAST Search History

L10	696	allocat$4 near3 (first adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L11	546	allocat$4 near3 (second adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L12	696	allocat$4 near3 (first adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L13	546	allocat$4 near3 (second adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L14	265	L12 same L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L15	6184	(sequential or consecutive) adj2 memory	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L16	265	L12 same L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L17	6184	(sequential or consecutive) adj2 memory	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L18	5	L16 and L17	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L19	316120	operat$4 adj system	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57

EAST Search History

L20	316120	operat$4 adj system	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L21	90	L20 and L16	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L22	3509	("without" or "no" or conceal$3 or bypass$4) near (operat$4 adj system)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L23	3509	("without" or "no" or conceal$3 or bypass$4) near (operat$4 adj system)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L24	2	L16 and L23	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L25	12	(Dennie near Shaun).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L26	696	allocat$4 near3 (first adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L27	546	allocat$4 near3 (second adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L28	696	allocat$4 near3 (first adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L29	546	allocat$4 near3 (second adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57

EAST Search History

L30	265	L28 same L29	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L31	3509	("without" or "no" or conceal$3 or bypass$4) near (operat$4 adj system)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L32	265	L28 same L29	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L33	3509	("without" or "no" or conceal$3 or bypass$4) near (operat$4 adj system)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L34	2	L32 and L33	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L35	6184	(sequential or consecutive) adj2 memory	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L36	6184	(sequential or consecutive) adj2 memory	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L37	5	L32 and L36	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L38	265	L28 same L29	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L39	316120	operat$4 adj system	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57

EAST Search History

L40	316120	operat$4 adj system	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L41	90	L40 and L32	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:57
L42	2	L32 and L33	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:58
L43	90	L40 and L32	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:58
L44	2	L42 and L43	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:58
L45	12	(Dennie near Shaun).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:58
L46	2	L42 and L43	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:58
L47	0	L45 and L46	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/08/14 08:58

Terms used
memory allocation sequential memory consecutive process operat system without bypass

Found **106** of **184,245**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 106 Result page: **1** 2 3 4 5 6 next

Relevance scale

1 External memory algorithms and data structures: dealing with **massive data**

Jeffrey Scott Vitter
June 2001 **ACM Computing Surveys (CSUR)**, Volume 33 Issue 2
Publisher: ACM Press

Full text available: pdf(828.46 KB) Additional Information: full citation, abstract, references, citings, index terms

Data sets in large applications are often too massive to fit completely inside the computers internal memory. The resulting input/output communication (or I/O) between fast internal memory and slower external memory (such as disks) can be a major performance bottleneck. In this article we survey the state of the art in the design and analysis of external memory (or EM) algorithms and data structures, where the goal is to exploit locality in order to reduce the I/O costs. We consider a varie ...

Keywords: B-tree, I/O, batched, block, disk, dynamic, extendible hashing, external memory, hierarchical memory, multidimensional access methods, multilevel memory, online, out-of-core, secondary storage, sorting

2 Scalable parallel allocation: Scalable locality-conscious multithreaded memory allocation

Scott Schneider, Christos D. Antonopoulos, Dimitrios S. Nikolopoulos
June 2006 **Proceedings of the 2006 international symposium on Memory management ISMM '06**
Publisher: ACM Press

Full text available: pdf(267.12 KB) Additional Information: full citation, abstract, references, index terms

We present Streamflow, a new multithreaded memory manager designed for low overhead, high-performance memory allocation while transparently favoring locality. Streamflow enables low over-head simultaneous allocation by multiple threads and adapts to sequential allocation at speeds comparable to that of custom sequential allocators. It favors the transparent exploitation of temporal and spatial object access locality, and reduces allocator-induced cache conflicts and false sharing, all using a un ...

Keywords: memory management, multithreading, non-blocking, shared memory, synchronization-free

3 The Alpine file system

M. R. Brown, K. N. Kolling, E. A. Taft
November 1985 **ACM Transactions on Computer Systems (TOCS)**, Volume 3 Issue 4
Publisher: ACM Press

Full text available: pdf(2.95 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Alpine is a file system that supports atomic transactions and is designed to operate as a service on a computer network. Alpine's primary purpose is to store files that represent databases. An important secondary goal is to store ordinary files representing documents, program modules, and the like. Unlike other file servers described in the literature, Alpine uses a log-based technique to implement atomic file update. Another unusual aspect of Alpine is that it performs all commu ...

4 Cache coherence in large-scale shared-memory multiprocessors: issues and comparisons

David J. Lilja
September 1993 **ACM Computing Surveys (CSUR)**, Volume 25 Issue 3
Publisher: ACM Press
Full text available: pdf(3.12 MB) Additional Information: full citation, references, citings, index terms

5 System-level power optimization: techniques and tools

Luca Benini, Giovanni de Micheli
April 2000 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 5 Issue 2
Publisher: ACM Press

Full text available: pdf(385.22 KB) Additional Information: full citation, abstract, references, citings, index terms

This tutorial surveys design methods for energy-efficient system-level design. We consider electronic sytems consisting of a hardware platform and software layers. We consider the three major constituents of hardware that consume energy, namely computation, communication, and storage units, and we review methods of reducing their energy consumption. We also study models for analyzing the energy cost of software, and methods for energy-efficient software design and compilation. This survery ...

6 Trace-driven memory simulation: a survey

Richard A. Uhlig, Trevor N. Mudge
June 1997 **ACM Computing Surveys (CSUR)**, Volume 29 Issue 2
Publisher: ACM Press

Full text available: pdf(636.11 KB) Additional Information: full citation, abstract, references, citings, index terms, review

As the gap between processor and memory speeds continues to widen, methods for evaluating memory system designs before they are implemented in hardware are becoming increasingly important. One such method, trace-driven memory simulation, has been the subject of intense interest among researchers and has, as a result, enjoyed rapid development and substantial improvements during the past decade. This article surveys and analyzes these developments by establishing criteria for evaluating trac ...

Keywords: TLBs, caches, memory management, memory simulation, trace-driven simulation

7 Fortran 8X draft

Loren P. Meissner
December 1989 **ACM SIGPLAN Fortran Forum**, Volume 8 Issue 4
Publisher: ACM Press
Full text available: pdf(21.36 MB) Additional Information: full citation, abstract, index terms

Standard Programming Language Fortran. This standard specifies the form and establishes the interpretation of programs expressed in the Fortran language. It consists of the specification of the language Fortran. No subsets are specified in this standard. The previous standard, commonly known as "FORTRAN 77", is entirely contained within this standard, known as "Fortran 8x". Therefore, any standard-conforming FORTRAN 77 program is standard conforming under this standard. New features can b ...

8 Disk cache—miss ratio analysis and design considerations
Alan J. Smith
August 1985 **ACM Transactions on Computer Systems (TOCS)**, Volume 3 Issue 3
Publisher: ACM Press
Full text available: pdf(3.13 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The current trend of computer system technology is toward CPUs with rapidly increasing processing power and toward disk drives of rapidly increasing density, but with disk performance increasing very slowly if at all. The implication of these trends is that at some point the processing power of computer systems will be limited by the throughput of the input/output (I/O) system. A solution to this problem, which is described and evaluated in this paper, is disk cache

9 File servers for network-based distributed systems
Liba Svobodova
December 1984 **ACM Computing Surveys (CSUR)**, Volume 16 Issue 4
Publisher: ACM Press
Full text available: pdf(4.23 MB) Additional Information: full citation, references, citings, index terms, review

10 Dataflow machine architecture
Arthur H. Veen
December 1986 **ACM Computing Surveys (CSUR)**, Volume 18 Issue 4
Publisher: ACM Press
Full text available: pdf(3.19 MB) Additional Information: full citation, abstract, references, citings, index terms

Dataflow machines are programmable computers of which the hardware is optimized for fine-grain data-driven parallel computation. The principles and complications of data-driven execution are explained, as well as the advantages and costs of fine-grain parallelism. A general model for a dataflow machine is presented and the major design options are discussed. Most dataflow machines described in the literature are surveyed on the basis of this model and its associated technology. F ...

11 HFS: a performance-oriented flexible file system based on building-block compositions
Orran Krieger, Michael Stumm
August 1997 **ACM Transactions on Computer Systems (TOCS)**, Volume 15 Issue 3
Publisher: ACM Press
Full text available: pdf(383.87 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The Hurricane File System (HFS) is designed for (potentially large-scale) shared-memory

multiprocessors. Its architecture is based on the principle that, in order to maximize performance for applications with diverse requirements, a file system must support a wide variety of file structures, file system policies, and I/O interfaces. Files in HFS are implemented using simple building blocks composed in potentially complex ways. This approach yields great flexibility, allowing an application ...

Keywords: customization, data partitioning, data replication, flexibility, parallel computing, parallel file system

12 Characterizing the caching and synchronization performance of a multiprocessor operating system

Josep Torrellas, Anoop Gupta, John Hennessy

September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems ASPLOS-V**, Volume 27 Issue 9

Publisher: ACM Press

Full text available: pdf(1.52 MB) Additional Information: full citation, references, citings, index terms

13 Fast detection of communication patterns in distributed executions

Thomas Kunz, Michiel F. H. Seuren

November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

Publisher: IBM Press

Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

14 Specifying Java thread semantics using a uniform memory model

Yue Yang, Ganesh Gopalakrishnan, Gary Lindstrom

November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande**

Publisher: ACM Press

Full text available: pdf(202.03 KB) Additional Information: full citation, abstract, references, index terms

Standardized language level support for threads is one of the most important features of Java. However, defining the Java Memory Model (JMM) has turned out to be a major challenge. Several models produced to date are not as easily comprehensible and comparable as first thought. Given the growing interest in multithreaded Java programming, it is essential to have a sound framework that would allow formal specification and reasoning about the JMM.This paper presents the Uniform Memory Model (UMM), ...

Keywords: Java, compilation, memory models, threads, verification

15 Instruction prefetching of systems codes with layout optimized for reduced cache misses

Chun Xia, Josep Torrellas

May 1996 **ACM SIGARCH Computer Architecture News , Proceedings of the 23rd**

annual international symposium on Computer architecture ISCA '96, Volume 24 Issue 2
Publisher: ACM Press

Full text available: pdf(1.65 MB) Additional Information: full citation, abstract, references, citings, index terms

High-performing on-chip instruction caches are crucial to keep fast processors busy. Unfortunately, while on-chip caches are usually successful at intercepting instruction fetches in loop-intensive engineering codes, they are less able to do so in large systems codes. To improve the performance of the latter codes, the compiler can be used to lay out the code in memory for reduced cache conflicts. Interestingly, such an operation leaves the code in a state that can be exploited by a new type of ...

16 Synchronization models: Composable memory transactions

Tim Harris, Simon Marlow, Simon Peyton-Jones, Maurice Herlihy
June 2005 **Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming**
Publisher: ACM Press
Full text available: pdf(239.37 KB) Additional Information: full citation, abstract, references, index terms

Writing concurrent programs is notoriously difficult, and is of increasing practical importance. A particular source of concern is that even correctly-implemented concurrency abstractions cannot be composed together to form larger abstractions. In this paper we present a new concurrency model, based on *transactional memory*, that offers far richer composition. All the usual benefits of transactional memory are present (e.g. freedom from deadlock), but in addition we describe new modular fo ...

Keywords: locks, non-blocking algorithms, transactions

17 Memory access scheduling

Scott Rixner, William J. Dally, Ujval J. Kapasi, Peter Mattson, John D. Owens
May 2000 **ACM SIGARCH Computer Architecture News , Proceedings of the 27th annual international symposium on Computer architecture ISCA '00**, Volume 28 Issue 2
Publisher: ACM Press

Full text available: pdf(181.84 KB) Additional Information: full citation, abstract, references, citings, index terms

The bandwidth and latency of a memory system are strongly dependent on the manner in which accesses interact with the "3-D" structure of banks, rows, and columns characteristic of contemporary DRAM chips. There is nearly an order of magnitude difference in bandwidth between successive references to different columns within a row and different rows within a bank. This paper introduces memory access scheduling, a technique that improves the performance of ...

18 Parallelizing nonnumerical code with selective scheduling and software pipelining

Soo-Mook Moon, Kemal Ebcioğlu
November 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 6
Publisher: ACM Press

Full text available: pdf(543.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Instruction-level parallelism (ILP) in nonnumerical code is regarded as scarce and hard to exploit due to its irregularity. In this article, we introduce a new code-scheduling technique for irregular ILP called "selective scheduling" which can be used as a component for superscalar and VLIW compilers. Selective scheduling can compute a wide set of independent operations across all execution paths based on renaming and forward-

substitution and can compute availab ...

Keywords: VLIW, global instruction scheduling, instruction-level parallelism, software pipelining, speculative code motion, superscalar

19 Functional-join processing

R. Braumandl, J. Claussen, A. Kemper, D. Kossmann

February 2000 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 8 Issue 3-4

Publisher: Springer-Verlag New York, Inc.

Full text available: pdf(486.22 KB) Additional Information: full citation, abstract, citings, index terms

Inter-object references are one of the key concepts of object-relational and object-oriented database systems. In this work, we investigate alternative techniques to implement inter-object references and make the best use of them in query processing, i.e., in evaluating functional joins. We will give a comprehensive overview and performance evaluation of all known techniques for simple (single-valued) as well as multi-valued functional joins. Furthermore, we will describe special *order-preser* ...

Keywords: *Functional join, Logical OID, Object identifier, Order-preserving join, Physical OID, Pointer join, Query processing*

20 Energy-aware design of embedded memories: A survey of technologies, architectures, and optimization techniques

Luca Benini, Alberto Macii, Massimo Poncino

February 2003 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 2 Issue 1

Publisher: ACM Press

Full text available: pdf(288.44 KB) Additional Information: full citation, abstract, references, index terms

Embedded systems are often designed under stringent energy consumption budgets, to limit heat generation and battery size. Since memory systems consume a significant amount of energy to store and to forward data, it is then imperative to balance power consumption and performance in memory system design. Contemporary system design focuses on the trade-off between performance and energy consumption in processing and storage units, as well as in their interconnections. Although memory design is as ...

Keywords: Embedded systems, embedded memories, integration, memories, nonvolatile, system-on-a-chip, volatile

IEEE Xplore® RELEASE 2.1

Welcome United States Patent and Trademark Office

Mon, 14 Aug 2006, 9:12:09 AM EST

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Search Query Display

Recent Search Queries

#1 ((memory allocation)<in>metadata)

#2 ((without or no or conceal* or bypass*) and (operating system)<IN>metadata)

#3 (allocat* and (first process)<IN>metadata)

#4 ((sequential or consecutive) and memory<IN>metadata)

#5 (shar* and (memory thread*)<IN>metadata)

#6 (((memory allocation)<in>metadata)) <AND> (((without or no or conceal* or bypass*) and (operating system)<IN>metadata))

#7 ((allocat* and (first process)<IN>metadata)) <AND> (((sequential or consecutive) and memory<IN>metadata))

#8 ((((memory allocation)<in>metadata)) <AND> (((without or no or conceal* or bypass*) and (operating system)<IN>metadata))) <AND> ((allocat* and (first process)<IN>metadata))

#9 ((shar* and (memory thread*)<IN>metadata)) <AND> ((((memory allocation) <in>metadata)) <AND> (((without or no or conceal* or bypass*) and (operating system)<IN>metadata)))

Indexed by Inspec®